

A Combined Process/Resource-oriented Approach to Shopfloor Simulation

Pyoungh Yol Jang*

Albert Jones**

Hyunbo Cho*

* Division of Mechanical and Industrial Engineering
Pohang University of Science and Technology
San 31 Hyoja, Pohang 790-784, Republic of Korea

** National Institute of Standards and Technology
Manufacturing Engineering Laboratory
Manufacturing Systems Integration Division
Gaithersburg, MD 20899

Abstract

Manufacturing simulation tools are used frequently to analyze proposed planning, scheduling, and control decisions that attempt to optimize some performance measure(s) for the shop. These shops exhibit a dynamic and complicated evolution caused by the concurrent flows of disparate parts following nonlinear process plans, machine failures, priority jobs, and variability in processing and transportation times. Capturing that evolution faithfully often requires modifications to the tools, the models created using the tools, or both. These modifications to the models are often difficult to make and prohibit timely and effective decisions.

This paper describes a new process and resource models-based intelligent simulator (PRISM), which simplifies those modifications. PRISM has four powerful capabilities. First, its process specification capability allows users to model nonlinear process plans that contain all the possible routings through the shop, decision-making rules associated with those routings, and precedence relationships among the processes in those routings. Second, a resource specification capability allows users to model properties of shopfloor resources and decision-making rules associated with those resources. Third, an automation capability that translates the process and resource models into a directed graph that the simulator engine uses directly. Fourth, a real-time analysis capability that simplifies integration of PRISM into many, existing, shopfloor control architectures.

Keywords: *Simulation, Nonlinear process plan, Directed graphs, Decision-making problem*

1. Introduction

Shops involved in the fabrication of discrete parts typically have a software system that must be able to plan, schedule, monitor, and control various machining and material handling devices. That system, called the shopfloor control system (SFCS), does this through a series of decisions that 1) ensures the completion of production orders and 2) optimizes one or more performance measures. In particular, once it receives the process plans for the parts to be produced, the SFCS makes decisions related to route selection, resource allocation, workpiece scheduling, instruction downloading, process monitoring, and error recovery [Cho, 1993]. Simulation has been used to support these decisions because it can model and analyze systems, like a manufacturing shop, that operate in an environment that is highly dynamic and unpredictable.

The first generation of simulation languages supported a process-oriented view of systems. They began to appear in the late 60s, and some are still in use today. They used a complex grammar to model a system as a network in which the nodes represent queues and the branches represent the paths connecting those queues. The second generation began to appear in late 80's. They include sophisticated, icon-based user interfaces and links to general-purpose programming languages such as C, C++, and VISUAL BASIC¹. They support a resource-oriented view -- the system is viewed as the specification and arrangement of resources that perform a number of different operations on entities as they flow through the resources. The process sequences are hidden within and across resources, as are the part characteristics. The capabilities of the first and second-generation simulation software for shopfloor control are summarized in Table I.

We believe that a simulator capable of process-oriented and resource-oriented views simultaneously would be beneficial for the following reasons. First, nonlinear process routings, which include both AND and OR branches, could be modeled more easily. Second, all of the important information about the parts and the resources would be visible, hence changeable, in the model. Third, the decisions made by the SFCS, which require up-to-date information about both the parts and the resources, could be better analyzed. Finally, a separation of the data required to run the model from the physical model of the shop itself could be achieved more easily. This would provide the SFCS with the ability to analyze many different scenarios before making a decision. The capabilities of the first- and second-generation simulation software for shopfloor control are summarized in Table I.

Table I. Characteristics of the first and second generation simulators

Characteristics	First generation	Second generation
Major viewpoint	Process-oriented view	Resource-oriented view
Process plan	Linear process plan	Not easy to represent
Decision-making problems	Not easy to handle	Not easy to handle
Material handling	Not easy	Easy
User interface	Not good	Good
Simplicity of usage	Not good	Good

¹ Certain commercial software products are identified in this paper. These products are for demonstration purposes only. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

The objective of the paper is to describe a new process and resource models-based intelligent simulator (PRISM). To this end, the paper describes the following: (1) the process model, which represents complex and flexible process plans for producing parts, (2) the resource model, which represents the characteristics and distributed relationships of various resources, (3) the simulator engine, which advances the simulation clock and manages the evolution of the simulation by investigating various pieces of information specified in the process and resource models, (4) the integration of PRISM into a real-time shopfloor controller as a decision-maker, and, (5) a comparison of PRISM with other simulation tools in terms of effectiveness and efficiency. The paper is organized as follows. Related work is presented in Section 2. The framework is given in Section 3. The process model is described in Section 4. The resource modeling method for SFCS resources is addressed in Section 5. The simulator engine is detailed in Section 6. A comparison with other simulation approaches is given in Section 7. Concluding remarks are presented in Section 8.

2. Some Related Work

2.1 Commercial Simulators

There are many commercial simulators on the market today; some of them are general-purpose simulators and some are designed specifically to model and analyze manufacturing systems. Most of the early simulators, including GPSS [Gorden, 1975], SIMAN [Pegden, 1982], SIMSCRIPT [Law *et al.*, 1984], and SLAM [Pritsker, 1986] are examples of process-oriented languages. These languages view systems as networks in which the nodes represent queues and the branches represent the paths connecting those queues. Each queue is connected to a process, and the emphasis is on the flow of entities through those processes -- not the processes themselves. This meant that the impact of the resources' properties and their distribution layout were hidden from the user. Consequently, those languages cannot support easily the dynamic decisions related to the concurrent movement of parts. Recent simulators, including WITNESS [AT&T, 1995], ProModel [Production, 1989], AutoMod [AutoSimulations, 1989], and SIMFACTORY [CACI, 1990], are resource-oriented languages. These languages view systems as a collection of resources that perform a number of different operations on entities as they flow through the resources. The emphasis is on the resources, not the entity flow. A manufacturing system can be modeled using either approach; the choice depends on the particular emphasis selected by the user.

2.2 Process Plan Representation

Traditional process plans are linear structures represented either by operation charts or routing sheets. These linear structures do not meet the needs of modern, more powerful SFCSs. To meet these needs, alternative, non-linear structures such as AND-OR graphs have been suggested [Chrysosolouris *et al.*, 1991], [Yeh *et al.*, 1991], [Hobbs *et al.*, 1992], [Kempenaers *et al.*, 1996] and [Kruth *et al.*, 1996]. [Cho *et al.*, 1994] developed computer-interpretable representations and parsers for these structures. They also described the limitations, in both structure and content, they encountered in trying to use these plans to resolve various decision-making problems and performing simulation. [Catron *et al.*, 1991] and [Shah *et al.*, 1995] proposed to use these AND-OR graphs as the basis for a natural hierarchy of plans to support various levels of planning and scheduling within the SFCS. IDEF3 is a tool for developing these types of plans [Mayer *et al.*, 1992], [Plaia *et al.*, 1995] and [KBSI, 1996]. IDEF3 can capture abstract knowledge about the dynamics of both business activities and manufacturing processes.

2.3 Academic Simulators

A number of academic researchers have embedded simulation tools in their SFCS. Son [Son, 2000] describes the Pennsylvania State University SFCS, which contains a simulation-based scheduler and a simulation-based controller. The former uses a preview simulation, which runs in the fast mode, to determine the schedule of jobs to be executed. The latter uses a simulation, which is tied to a real clock, to release those jobs at the appropriate time. Cho [Cho, 1993] used neural networks to identify candidate rules for multi-pass simulation analysis to determine the sequence of jobs at an automated workstation. At each decision point, the neural network generates candidate rules for five types of scheduling problems, and these rules are then evaluated through simulation. The benefits of real-time simulation in the control of manufacturing systems were first proposed by Davis and Jones [Davis *et al.*, 1988]. Davis and his colleagues have built upon those early results to develop extensive, real-time emulations of the control systems at several, real, manufacturing facilities [Davis *et al.*, 1996]. Drake et al. [Drake *et al.*, 1995] presented a framework for applying simulation models to on-line planning, scheduling, and control problems. Smith et al. [Smith *et al.*, 1994] examined the application of simulation as both decision maker and task generator. The decision maker determines what task should occur next; the task generator and sends that task to the system's execution software.

3. Framework of the PRISM

The concept for PRISM is shown in Figure 1. PRISM can be applied for all the levels of SFCS such as shop, cell, workstation, and equipment (the number of levels can be changed according to the specific SFCS) as shown in Table 2. In each

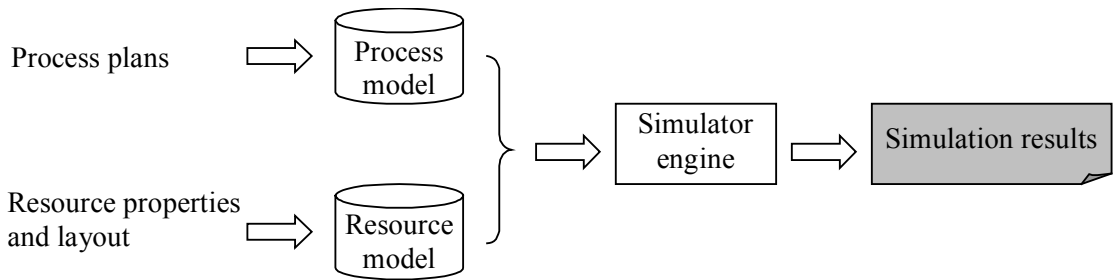


Figure 1. Concept for PRISM

level, the elements and decision-making problems of the process model and resource model are changed to reflect the nature of corresponding level. In addition, the PRISM can be applied for all the control architectures such as central, hierarchical, hybrid, and distributed. For the hierarchical and hybrid SFCS, it is necessary to aggregate the element of the process model and resource model from equipment to shop and to disaggregate the decision-making problems from shop to equipment.

Table 2. Roles of the PRISM for various control levels and control architectures

Level	Process Model		Resource Model		Applicable Control Architecture
	Elements of Process Plans	Decision-Making Problem	Elements of Resources	Decision-Making Problem	
<i>Shop</i>	Cell Process, AND/OR Junction	Cell selection, Part buffering, Process sequence, Path selection	Cell, Storage, Transport	Part selection, Transport selection	Central/Hierarchical/Hybrid
<i>Cell</i>	Workstation Process, AND/OR Junction	Workstation selection, Part buffering, Process sequence, Path selection	Workstation, Storage, Transport	Part selection, Transport selection	Central/Hierarchical/Hybrid
<i>Workstation</i>	Machine Process, AND/OR Junction	Machine selection, Part buffering, Process sequence, Path selection	Machine, Storage, Transport	Part selection, Transport selection	Distributed/Central/Hierarchical/Hybrid
<i>Equipment</i>	Cutting Process, AND/OR Junction	Cutting tool selection, Part buffering, Process sequence, Path selection	Cutting tool, Storage, Transport	Part selection, Transport selection	Distributed/Central/Hierarchical/Hybrid
Aggregation/Disaggregation	<i>Aggregation</i>	<i>Disaggregation</i>	<i>Aggregation</i>	<i>Disaggregation</i>	

4. The Process Model

When it arrives at the SFCS, a part carries a process plan that must specify the various processes to produce a finished part from the raw material. It must also contain temporal precedence relationships among these processes.

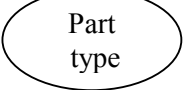
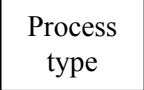


	Head	Process	Junction	Link
Symbol				
Type	Part types	Milling, Drilling, Turning, Chamfering, etc.	AND(A), OR(O)	None
Characteristics	Prefixed symbol for every process plan	Any process type that requires "duration"	Part flow logic for flexible nonlinear process plan	Temporal precedence among processes

Figure 2. Symbols used in the process model

In the past, a process plan was represented as a linear sequence of processes or resources. Recently, as noted above, many researchers have argued for allowing alternative processes and resources. We incorporate these alternatives and model the resulting process plan as an AND/OR graph in which a node is one of three kinds, *head*, *process*, *junction*, and an edge is denoted as *link*, as shown in Figure 2. The *head* node contains the information needed for the creation of a part for simulation, such as first arrival time, inter-arrival time distribution, and lot size of the associated part. The *process* node represents a single machining operation performed on a part, such as turning, milling, or drilling. This node contains the information contents on the service resources, such as machine tool, cutting tool, and fixture. The *junction* node denotes the part flow logic. There are AND junctions (displayed as "A") and OR junctions (displayed as "O"), each of which consists of fan-out and fan-in pairs. The symbol *link* represents the precedence relationship among processes and junctions in the process plan.

4.1 Head Symbol and Information

Each part that enters the shop has its own collection of simulation attributes that are managed by the symbol *head*. They may include some or all of part name, first arrival time, last arrival time, maximum number of arrivals, inter-arrival time, and lot size (see Figure 3). Each numerical attribute can be represented as either a constant number or a probability distribution function. This symbol must be attached to the head of the process model corresponding to a part.

Part Name	Part_1
First Arrival Time	10
Last Arrival Time	1000
Max. Arrival Num.	100
Int. Arrival Time	3
Lot Size	1

OK Cancel

Figure 3. Properties of the symbol *head*

Process Name	FACE_MILL
--------------	-----------

Resources	
Machine tool	M1
Cutting tool	C1
Fixture	F1
Processing Time	20

NC codes...
Add Alternative Resources...

Decision Rules	
Resource selection rule	Min. Processing Time
Part buffering rule	Ratio of Remaining Time

OK Cancel

Figure 4. Properties of the symbol *process*

4.2 Process Symbol and Information

The attributes of the *process* node are process name, resources, and decision rules as shown in Figure 4. The process name is simply an identifier. There are two resource types: primary (Numerically Controlled (NC) machine) and secondary (cutter, fixture, and so on). The processing time can be provided directly or estimated from the NC code. Two decision problems are possible at in every process step in the process plan: resource selection and part buffering. If alternative resources are specified for a process, the simulator must choose a primary and secondary resource set from the resource database. This is done using user-specified rules such as minimum processing time, minimum transport time, or minimum waiting time. For example, if the minimum-transport-time rule is selected, the simulator engine selects the machine tool located nearest to the current part's location. The part-buffering problem is associated with the determination of the next destination for the part when no resource specified for the next process is available. In that case, the simulator engine may (1) send the part to the buffer, (2) let it stay at the current location, or (3) send it to the material handler. An exemplary rule used to select one of the options is the ratio of remaining time to total machining time of the next process -- if this ratio is larger than a pre-specified value, the part can be moved to the buffer. Figure 5 illustrates the sequence of rules fired when the simulator engine has already finished process P1 and examines the next process P2.

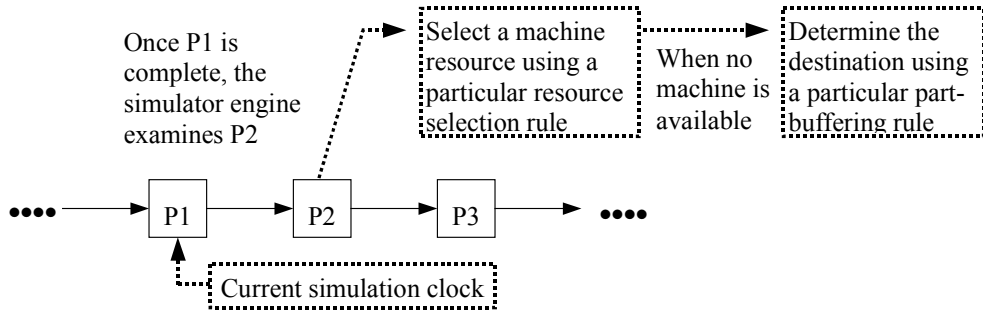


Figure 5. Sequence of fired rules after process P1 is finished

4.3 Junction Symbols and Information

The AND junction (A) means that all the processes between the starting A and ending A must be performed. In Figure 6a, processes P1, P2, and P3 lie between AND junctions. All of them must be done -- three of the six possible sequences are shown. The manufacturing efficiency, which is a function of setup time, tool changes, and fixture changes, will depend on which of these is selected.

The OR junction (O) means that one and only one of the processes or paths between the starting OR and the ending OR junctions should be selected and then executed. In Figure 6b, there are two possible choices: P1 followed by P2 or P3.

Realistic process models will contain many such junctions, some of which will be nested. Consider Figure 6c. The correct interpretation is that P1, P2, and one of P3 or P4 must be done. Several possible sequences are shown. In Figure 6, several examples are illustrated.

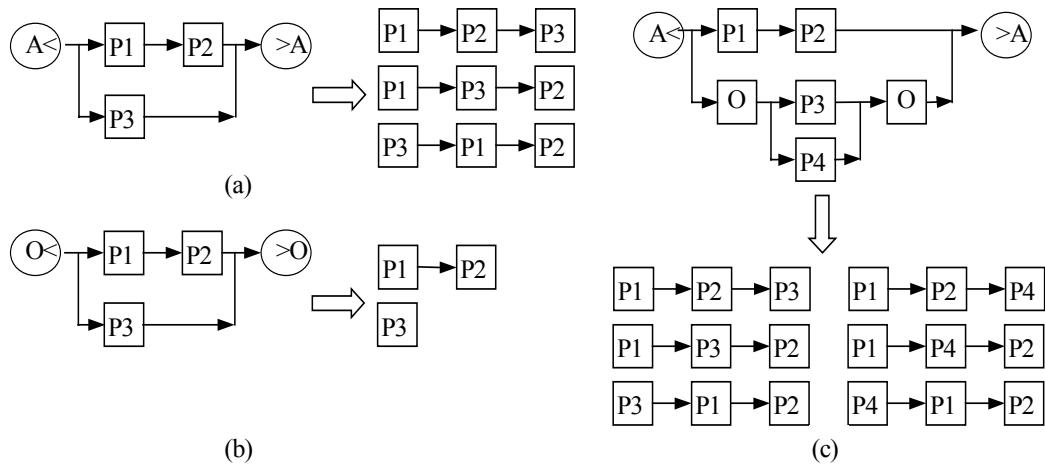


Figure 6. Examples of serialization of the processes surrounded with junctions

When the simulator engine hits the initial AND junction, it does not immediately sequence all of the ensuing nodes. Instead, it chooses only the next node in the sequence. It does this one-at-a-time selection until the final AND junction is reached. This procedure, called the process sequence procedure, uses both the current state information and the rules imbedded in the definition of an AND junction. For example, if the rule is "minimum traveling time", then the process selected is the one to be executed on the machine located nearest to the current

location.

When the simulator engine encounters an initial OR junction, it prioritizes all of the possible paths or processes, then selects the one with the highest priority. This procedure, called the path selection procedure, is also based on the current state and the rules imbedded in the OR junction. For example, the maximum-flexibility rule will select that path with the largest number of AND junctions, because that path has many processes that can be sequenced later.

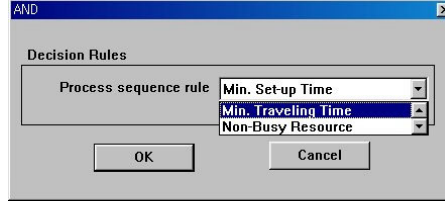


Figure 7. Properties of the symbol *AND* junction

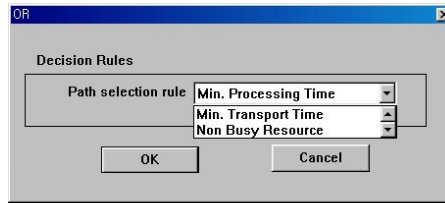


Figure 8. Properties of the symbol *OR* junction

The internal data structure of the process model consists of tables and sets. The tables store information about individual heads, processes, and junctions. The sets are candidates, alternatives, and order. The *candidate set* contains all executable processes for which no precedence exists. The process selection procedure chooses from this set. The *alternative set* contains all the paths between the corresponding OR junctions. The path selection procedure chooses from this set. The *order set* contains the serialized processes. While every process in a linear process plan must be in a process order set, only the first process in the set can be selected. We note that a given set might contain other sets as members. An exemplary process model and its corresponding set are shown in Figure 9.

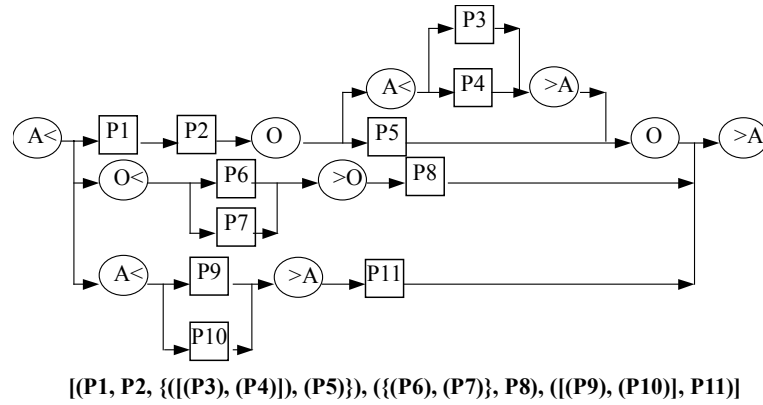


Figure 9. Sample process model and its corresponding set

4.4 Example Part and Its Process Model

An example of a produced part and its related process model, 10 processes, 2 junctions, and 1 head, are illustrated in Figure 10. The specific features to be machined and their related processes are also represented. As for temporal precedence relationships, feature A must be initially removed. Others can be removed in several different sequences. Features B, C, D, E, F, and stepped pocket feature G can be machined in any sequence, but only after feature A is machined. Hence, they can be grouped with AND junctions. The stepped pocket feature can be machined in two alternative ways: pocket feature G is removed and then feature H is drilled, or feature I is drilled and then pocket feature J is removed.

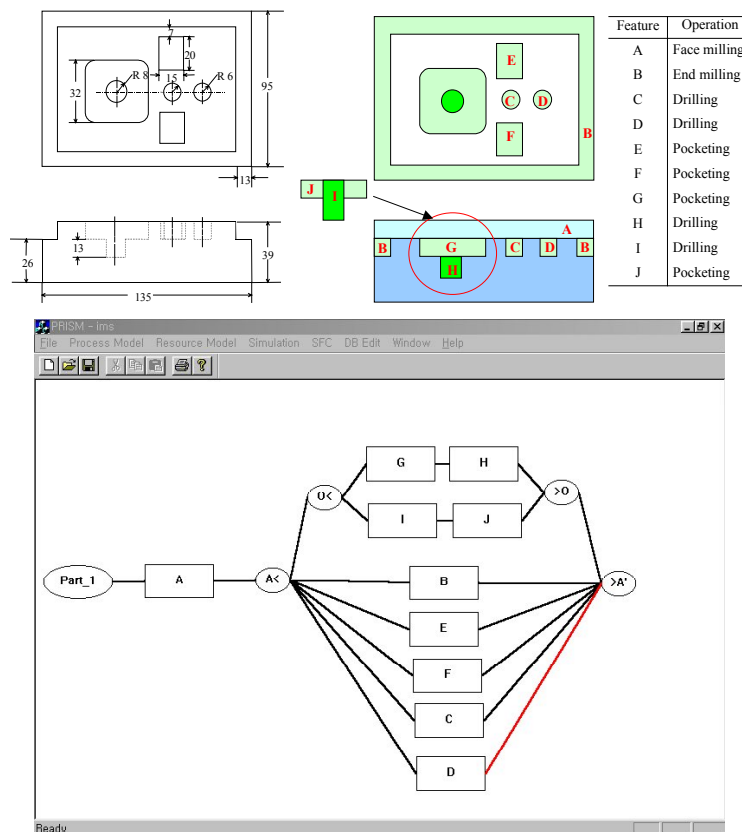


Figure 10. Example part and its process model

5. Resource Modeling Method

A process model contains no information about the resources on which processes are executed. PRISM represents and stores this information, which includes such characteristics as resource type, resource properties, resource location, number of resources, and distances between pairs of resources, represents and stores this information in a separate model called the Resource Model. Many commercial simulators contain such a model, but PRISM's model has four major advantages. First, each resource instance is associated with a specific decision-making problem, such as sequencing or scheduling, and has embedded rules for solving that problem. Second, the machine-tool resource includes an NC interpreter

to estimate the processing time for each feature. Third, tool and fixture resources are introduced to allow more detailed optimization. Fourth, robot and automated storage and retrieval systems (AS/RS) are included to allow development of more accurate models of the shop.

5.1 Resource Types and Interactions

There are three resource classes: processing, storage, and transport as shown in Figure 11. A processing resource, typically a machine tool or a human being, performs various machining or inspection processes. A storage resource, typically buffer or a storage facility, houses parts and materials for varying amounts of time. A transport resource, typically a material handler or a material transporter, moves parts and materials among the other resources. Material handlers are dedicated to one or more resources located close together; they move parts and materials among those resources only -- a robot is the most common example. Material transporters are not dedicated to any particular resources; they move parts and materials over longer distances among many resources -- AGVs and conveyors are the most common examples.

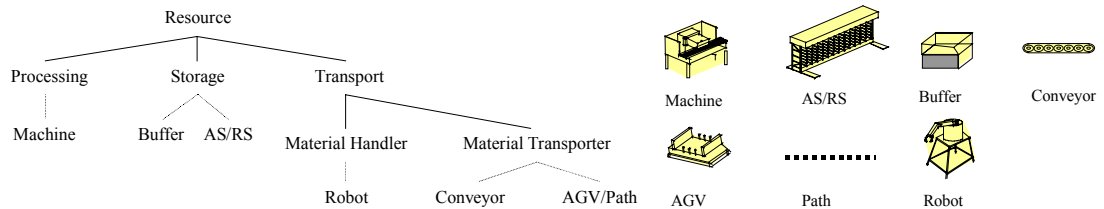
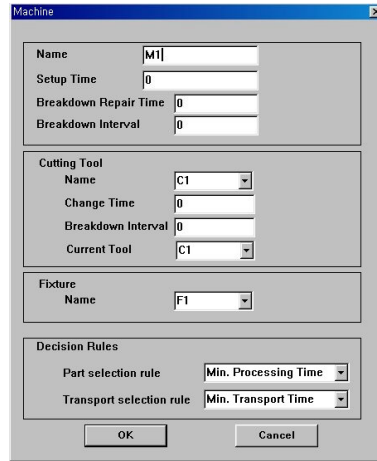


Figure 11. Types and symbols of resources

Since all resources must work together to manufacture parts, we must specify their interactions. We assume that all interactions involve a material handler, because it alone picks up from one resource and puts down at another resource. Therefore, information on all interactions among resources can be specified in the material handler resource. Because of this, we can represent the entire resource model as a directed graph in which an edge represents a material handler and each node represents one of the other resources.

5.2 Processing Resource Symbols and Information

A processing resource executes each "machine" process step in the process model. Its properties are shown in Figure 12, including any secondary resources (see above). Since resource may have a breakdown, breakdown intervals and repair times are also included. There are also two decision-making



The image shows a 'Machine' properties dialog box with the following fields and values:

- Name: M1
- Setup Time: 0
- Breakdown Repair Time: 0
- Breakdown Interval: 0
- Cutting Tool Name: C1
- Change Time: 0
- Breakdown Interval: 0
- Current Tool: C1
- Fixture Name: F1
- Decision Rules:
 - Part selection rule: Min. Processing Time
 - Transport selection rule: Min. Transport Time

Buttons: OK, Cancel

Figure 12. Properties of the *machine*

problems: transport selection and part selection. The transport selection problem is to determine the transport path and all required transport resources needed to move the part from its current location to its next location. A path is chosen from the available candidates identified from the directed graph. The choice is made using the imbedded rules such as "Minimum transport time" or "Non-busy transport". The part selection problem is to choose the next part to be processed from among those waiting in the buffer or other storage locations. Many rules can be used for resolving the part selection problem, such as "Minimum processing time", "Minimum transport time", or "Minimum waiting time". Figure 13 illustrates the procedure for part selection.

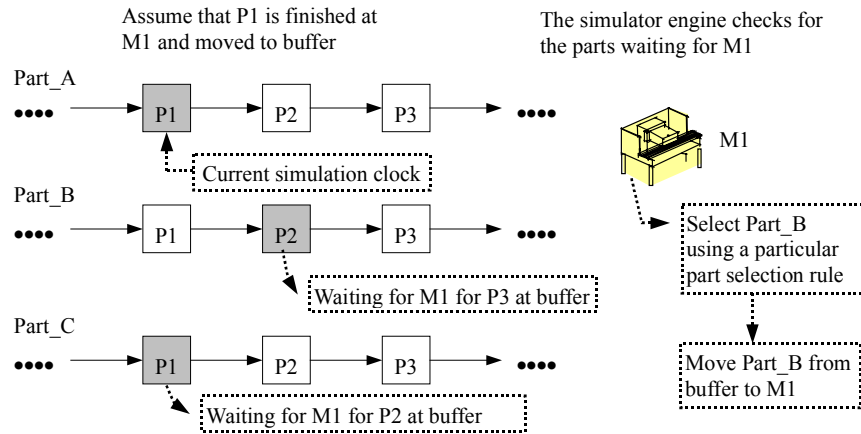


Figure 13. Conceptual situation of the part selection problem

5.3 Storage Resource Symbols and Information

There are two types of storage resources: AS/RS and buffer. An *AS/RS* resource is an active device that stores and retrieves in-process parts and raw materials automatically. A *buffer* resource is a passive device that can only store parts and materials. The properties of the AS/RS and buffer are shown in Figure 14.

The figure shows two side-by-side dialog boxes. The left dialog is titled 'AS/RS' and contains the following fields: Name (AS1), Quantity (1), Capacity (1), Breakdown Interval (0), Breakdown Repair Time (0), Time to Receive (2), Time to Retrieve (2), and a Service Resources section with a dropdown menu showing 'M2'. The right dialog is titled 'Buffer' and contains: Name (B1), Quantity (1), Capacity (1), and a Service Resources section with a dropdown menu showing 'M1'. Both dialogs have 'OK' and 'Cancel' buttons at the bottom.

Figure 14. Properties of the *AS/RS* and *buffer*

5.4 Transport Resource Symbols and Information

There are three transport resources: AGV, robot, and conveyor. The *AGV* resource definition is shown in Figure 15, which also shows the definition of “path”. A path joins storage resources. Routes are defined by linking these paths together using the Name and Next Path identifiers. The *AGV* resource is associated with two decision-making problems: AGV location and part selection. AGV location is to determine the next destination when an AGV becomes idle. Several rules can be applied including "Expected urgent loading", "Go to home position", and "Stop". For example, the Expected-urgent-loading rule implies that the AGV moves to the load station from which the simulator expects an urgent loading request to come. A part selection problem is the same as that of the machine tool. The properties of path and AGV are shown in Figure 15.

The figure shows two side-by-side dialog boxes. The left dialog is titled 'Path' and contains: Name (P1), Length (10), Max Speed (10), Number of AGVs Allowed (1), Next Path (P2), Loading Capacity (1), and Unloading Capacity (1). The right dialog is titled 'AGV' and contains: Name (AGV1), Capacity of Parts (1), Breakdown Repair Time (0), Breakdown Interval (0), Speed in Load (5), Speed in Unload (4), Home Position (P1), Park Zone in Home Position (Front), and a Decision Rules section with two dropdown menus: 'Part selection rule' (Busiest Loading Station) and 'AGV location rule' (Go to Home Position). Both dialogs have 'OK' and 'Cancel' buttons at the bottom.

Figure 15. Properties of the *path* and *AGV*

The *robot* resource provides loading and unloading services for both process and storage resources. The robot is associated with two decision-making problems: robot location and part selection. The robot location problem is to determine the next position of the robot when it becomes idle. There are several rules used for resolving the problem such as "Minimum remaining processing time", "Go to home position", and "Stop". For example, the Minimum-remaining-processing-time rule implies that the robot moves to the machine that has the minimum remaining time to finish its ongoing process. A part selection problem is the same as that of the machine tool.

The *conveyor* resource transports the parts and materials from one place to another, and it has cycle time to transport and length in parts. ‘Length in Parts’ is the maximum number of parts the conveyor can carry. The properties of the robot and conveyor are shown in Figure 16.

The screenshot displays two overlapping dialog boxes. The 'Robot' dialog box is in the foreground, featuring a 'Name' field with 'R1', a 'Home Position' dropdown set to 'M1', and three input fields for 'Breakdown Repair Time' (0), 'Breakdown Interval' (0), and 'Pick Part Time' (0). Below these is a 'Service Resources' section with 'From Resource' (M1) and 'To Resource' (M2) dropdowns, and two input fields for 'Cycle Time with Part' (10) and 'Cycle Time with No Part' (9). The 'Decision Rules' section includes 'Part selection rule' (Min. Waiting Time) and 'Robot Location Rule' (Go to Home Position). At the bottom are 'OK' and 'Cancel' buttons. The 'Conveyor' dialog box is partially visible behind it, showing 'Name' (C1), 'Quantity' (1), 'Cycle Time' (4), 'Breakdown Interval' (0), 'Breakdown Repair Time' (0), and 'Length in Parts' (10), with its own 'OK' and 'Cancel' buttons.

Figure 16. Properties of the *robot* and *conveyor*

5.5 Example

An exemplary resource model of a particular shop is shown in Figure 17 with its converted directed graph. It consists of 3 machines, 3 robots, an AS/RS, a Buffer, and an AGV. To create the concerned resource model, the user can pick up appropriate icons from the menu and arrange them on the working area. The user can modify the properties of each resource instance by clicking twice the corresponding icon.

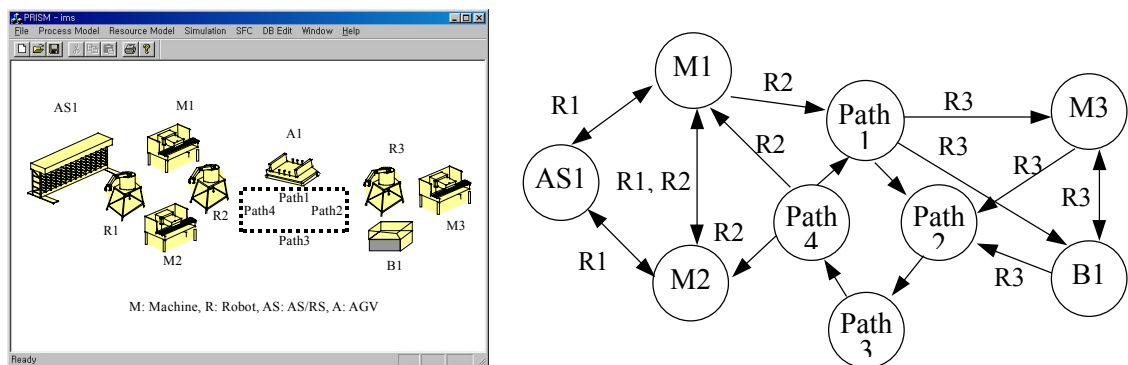


Figure 17. Exemplary resource model and its converted directed graph.

6 Process and Resource Models-based Simulator Engine

6.1 The Flow of Simulator Engine

The simulator engine of PRISM has five modules: initializer, clock manager, event manager, report generator, and kernel. The initializer sets the preliminary values for the simulation clock, system's state, statistical counters, and event list. The clock manager selects the next event to be fired from the event list and pushes forward the simulation clock to the time of that event [Law *et al.*, 1991]. The event manager updates the system's state and

statistical counters, generates future events, and adds them to the event list. The report generator produces the output results when the simulation completes. The PRISM kernel manages the other modules. It invokes the initializer and then repeatedly invokes the clock manager and the event manager until the end of the simulation.

While a simulation is running, the simulator engine retrieves the information needed to generate future events and make decisions from the process and resource models. The detail information flow directed to the simulator engine from two models is illustrated in Figure 18.

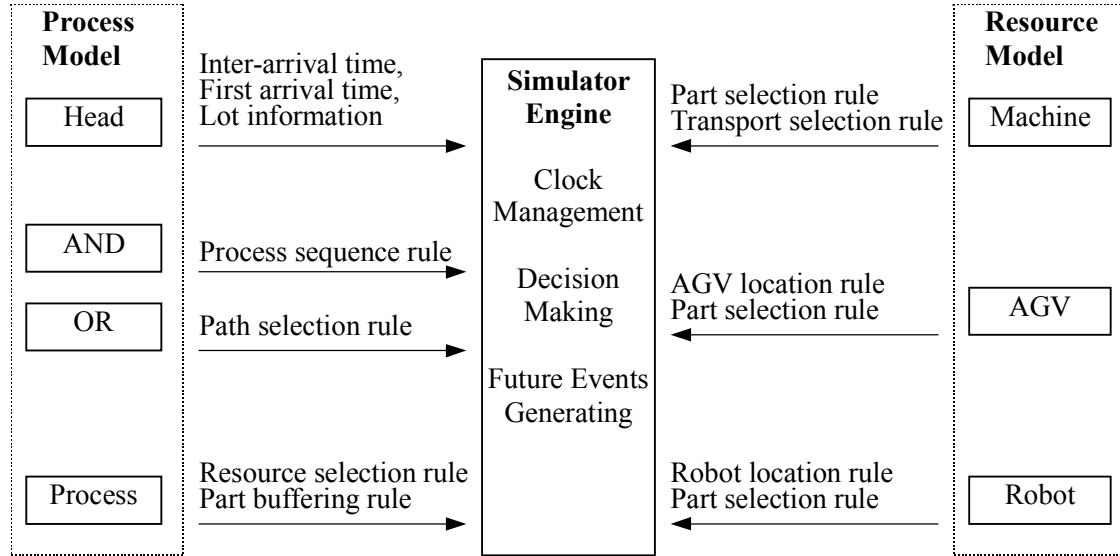


Figure 18. Information flow directed to the simulator engine

6.2 The Procedure of Simulation

PRISM's user interface has four different windows: process model, resource model, simulator engine, and statistics results. The process and resource model windows are used to construct the process and resource models. The simulator engine windows shows the evolution of the simulation as the clock advances. The statistics result window shows the statistics of various simulation-related variables. The exemplary process and resource model is shown in Figure 19. The process model consists of one head, seven processes, and four junctions. The resource model consists of two machines and one robot.

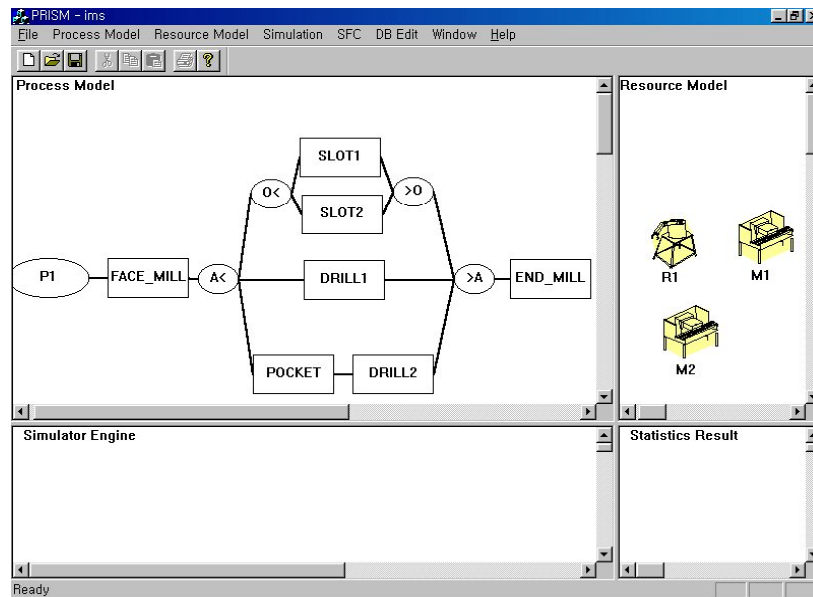


Figure 19. Constructed process and resource model

After running the simulation, the AND and OR junctions are resolved by the pre-specified process sequence and path selection rules. The resulting sequence of processes - FACE_MILL->SLOT1->POCKET->DRILL1->DRILL2->END_MILL - is shown in the top left window in Figure 20. The top of the simulator status window, the bottom left window in Figure 20, shows the transport matrix that was derived from the directed graph of the resource model.

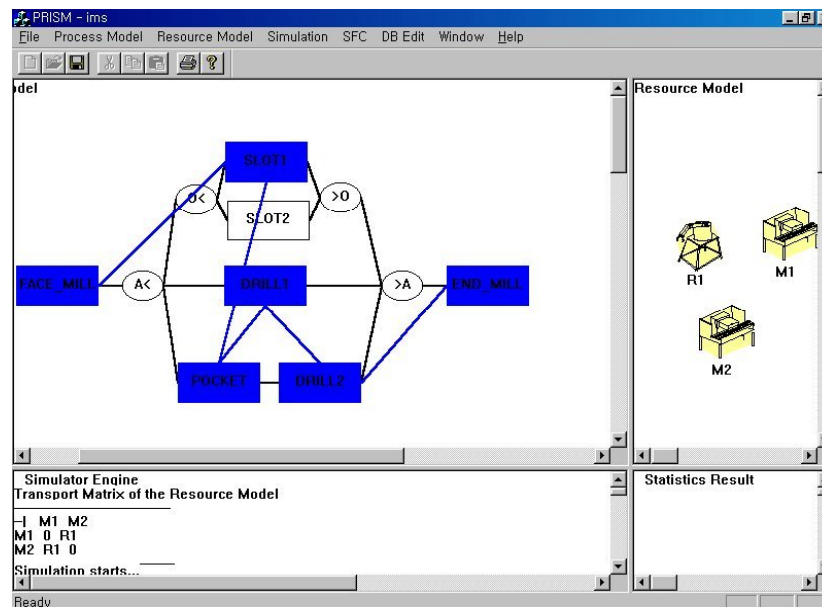


Figure 20. Linearized process model

A detailed log of the progress and status of the simulation of a single job is shown in Figure 21. The job arrives at time 10 and proceeds to the FACE_MILL process, which is executed on machine M1. At time 22, there are four candidate processes: SLOT1, SLOT2, DRILL1, and POCKET. The simulator reads the process and resource models and uses the pre-specified rules to select the next process and resource -- in this case, process

SLOT1 on machine M1. This continues until time 82 when the simulation ends.

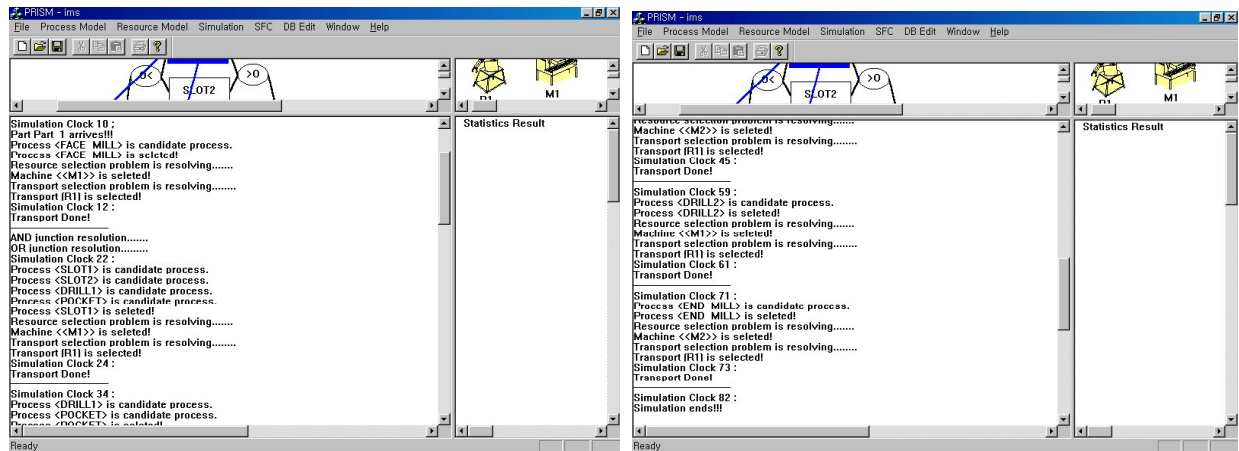


Figure 21. Progress and status of simulator engine

7 Comparison with Other Simulators

Because it integrates both a process view and a resource view, PRISM has several advantages over languages that support only one of these views:

- 1) Grammar: PRISM has a very easy grammar that simplifies the construction and maintenance of a simulation model. You just pick and place process and resource icons, and then define the properties of icons.
- 2) Model dynamics: PRISM represents processes and resources separately, which simplifies the construction of process plans and the capture of resource properties and shop layout. Languages that implement a process view only, must represent process plans and resources together in the same symbols or blocks. In order to represent the layout of resources, the user must define the layout in a separate window just for the animation.
- 3) Part flow characteristics: PRISM captures part flow characteristics in the directed graph, which is generated automatically from the process and resources models. In languages that support a resource view only, the logic to control part flow is hidden in the resource descriptions.
- 4) Nonlinear process plan: PRISM is designed to define non-linear process plans, those with alternatives for both processes and resources, easily. In process-oriented languages, such plans must be implemented in user code, which is written in a commercial programming language or a language provided with the package. In some cases, it is procedural code attached as attributes to a part entity; and, in other cases, the plans are stored in external databases. In resource-oriented languages, each resource has its own PUSH and PULL logic to receive and send parts. For example, Figure 22 shows a simple process plan and shopfloor with 5 processes and 5 machines. If each process can be executed on any machine, the arrows show the possible movements of parts among the resources. In this case, the number of possible paths through the shop is $5! = 120$ and number of possible resource selections is $5^5 = 3125$. Therefore, the total number of combinations is 375000. It needs a lot of user-coded algorithms to represent such a number of part sequence logic in the resource model only.

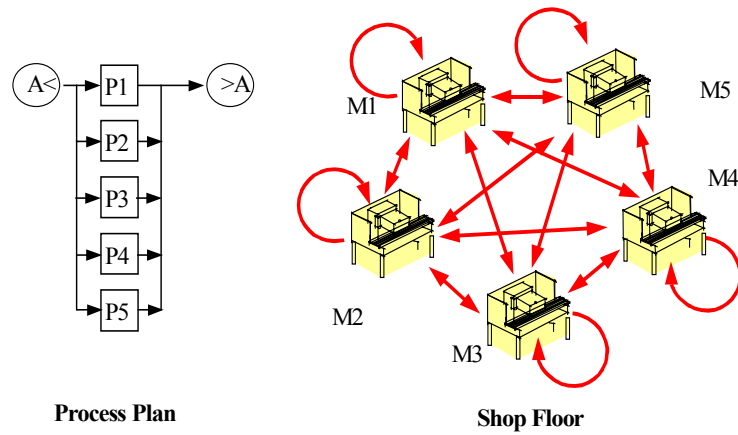


Figure 22. Exemplary process and resource model

8 Conclusion

This paper describes PRISM, a new simulation paradigm for SFCS. This paradigm integrates process models and resource models. The process model includes information about the processes, and all possible paths through those processes, for producing parts. The resource model includes information about the characteristics and distributed relationships of various resources. The simulator engine advances the simulation clock and manages the events using the information in these models. In particular, the simulator engine utilizes the decision-making rules to solve various selection problems associated with the processes and resources.

The major contribution of the paper is to enable the system analysts to perform off-line simulations rapidly and effectively to estimate the impacts of various control strategies of a shopfloor. The analysts have only to capture the process model of produced parts and to construct the resource model of shop resources' properties and layout. In the future, PRISM can be embedded into a real-time shopfloor controller as a decision-maker if the self-generated events are replaced by the monitoring events coming from the shopfloor and it has the decision generation, analysis, and optimization capability.

Acknowledgement

This work was supported by grant (No. 1999-2-315-002-3) from the Interdisciplinary Research Program of the KOSEF.

References

- [1] AT&T ISTEEL, *Witness User Manual, Release 7.0*, Visual Interactive Systems, UK, 1995.
- [2] AutoSimulations, Inc., *AutoMod II Users Manual*, Bountiful, Utah, 1989.
- [3] CACI Products Company, *SIMFACTORY II.5 User's and Reference Manual*, Version 2.0, LaJolla, Calif., 1990.
- [4] Catron, B. A. and Ray, S., "ALPS: a language for process specification", *International Journal of Computer Integrated Manufacturing*, Vol. 4, No. 2, pp. 105-113, 1991.

- [5] Cho, H., *An Intelligent Workstation Controller for Computer Integrated Manufacturing*, Ph. D. Dissertation, Texas A&M University, 1993.
- [6] Cho, H., Derebail, A., Hale, T. and Wyak, R. A., "A formal approach to integrating computer aided process planning and shop floor control", *ASME Transactions: Journal of Engineering for Industry*, Vol. 116, No. 1, pp. 106-116, 1994.
- [7] Chrysosolouris, G., Dick, K. and Lee, M., "Short interval scheduling for discrete parts manufacturing", *International Journal of Computer Integrated Manufacturing*, Vol. 4, pp. 157-168, 1991.
- [8] Davis, W. and Jones, A. T., "A real-time production scheduler for a stochastic manufacturing environment", *International Journal of Computer Integrated Manufacturing*, Vol. 1, No. 2, pp. 101-112, 1988.
- [9] Davis, W., Macro, J., Brook, A., Lee, M., and Zhou, G., "Developing a Real-Time Emulation/Simulation Capability for the Control Architecture to the RAMP FMS", *Proceedings of the 1996 Winter Simulation Conference*, Coronado, CA, USA, pp. 171-178, December, 1996.
- [10] Drake, G. R., Smith, J. S., Peters, B. A., "Simulation as a planning and scheduling tool for flexible manufacturing systems", *Proceedings of the 1995 Winter Simulation Conference*, pp. 805-812, 1995.
- [11] Gordon, G., *The Application of GPSS V to Discrete System Simulation*, Prentice-Hall, Englewood Cliffs, N. J., 1975.
- [12] Hobbs, S. and Steel, R., "Process planning tools", *ESPRIT Workshop on Flexibility through Integrated Design : Process Planning and Scheduling*, Senlis/Paris, France, 1992.
- [13] Kempenaers, J., Pinte J., Detand, J., and Kruth, J. P., "A collaborative process planning and scheduling system", *Advances in Engineering Software*, Vol. 25, No. 1, pp. 3-8, 1996.
- [14] Kruth, J. P., Detand, J., Zeir, G., Kempenaers J., and Pinte, J., "Methods to improve the response time of a CAPP system that generates non-linear process plans", *Advances in Engineering Software*, Vol. 25, No. 1, pp. 9-17, 1996.
- [15] KBSI, *PROSIM User's Manual*, Version 2.2.1, Knowledge Based Systems Inc., 1996.
- [16] Law, A. M. and Larmey, C. S., *Introduction to Simulation Using SIMSCRIPT II.5*, CACI Products Company, La Jolla, Calif., 1984.
- [17] Law, A. M. and Kelton, W. D., *Simulation Modeling and Analysis*, McGRAW-HILL, Singapore, 1991.
- [18] Mayer, R. J., Cullinane, T. P., deWitte, P. S., Knappenberger, W. B., Perakath, B., and Wells, M. S., *IDEF3 Process Description Capture Method Report*, KBSI, Texas, 1992.
- [19] Pegden, C. D., *Introduction to SIMAN*, Systems Modeling Corporation, Pennsylvania, 1982.
- [20] Plaia, A. and Carrie, A., "Application and assessment of IDEF3 – process flow description capture method", *International Journal of Operations & Production Management*, Vol. 15, No. 1, pp. 63-73, 1995.
- [21] Pritsker, A. A. B., *Introduction to Simulation and SLAMII*, Systems Publishing Corporation, 1986.
- [22] Production Modeling Corporation of Utah, *ProModel User's Manual*, Orem, Utah, 1989.
- [23] Shah, J. and Mantyla, M., *Parametric and feature-based CAD/CAM*, John Wiley, New York, 1995.
- [24] Smith, J. S., Wysk, R. A., Sturrok, D. T., Ramaswary, S. E., Smith, G. D., and Joshi, S. B., "Discrete Event Simulation for Shop Floor Control", *Proceedings of the 1994 Winter Simulation Conference*, pp. 962-969, 1994.
- [25] Son, Y. J., *Simulation Based Shop Floor Control: Automatic Model Generation and Control Interface*, Ph. D. Dissertation, Pennsylvania State University, 2000.
- [26] Yeh, C. H. and Fisher, G. W., "A structured approach to the automatic planning of machining operations for rotational parts based on computer integration of standard design and process data", *International Journal of Advanced Manufacturing Technology*, Vol. 6, pp. 285-298, 1991.